MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2016/2017

TDS 2111 – DATA STRUCTURES AND ALGORITHMS

(All sections / Groups)

21 OCT 2016
3.00 p.m – 5.00 p.m
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This Question paper consists of 7 pages (excluding cover page) with 4 Questions.
2. Attempt **ALL** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please write all your answers in the Answer Booklet provided

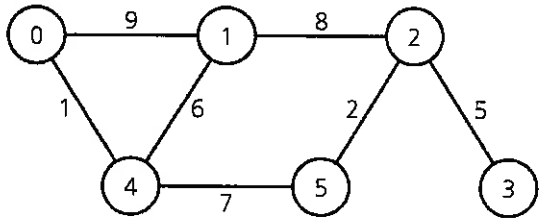
Question 1

(a) Distinguish between a data structure and an abstract data type (ADT).

[2 marks]

(b) Give the adjacency matrix for the following weighted graph.

[3 marks]



(c) Briefly explain the definition of ADT Stack?

[2 marks]

(d) What is the output from the following sequence of queue operations?

[3 marks]

```
#include<iostream>
#include<queue>
using namespace std;

int main()
{
queue<int> q;
int x = 5, y = 3;
q.push(8);
q.push(9);
q.push(y);
x = q.front();
q.pop();
q.push(18);
x = q.front();
q.push(22);
while(!q.empty())
{
    y = q.front();
    q.pop();
    cout << y << " ";
}
cout << x << endl;
system("PAUSE");
return 0;
}
```

Continued .....

Question 2

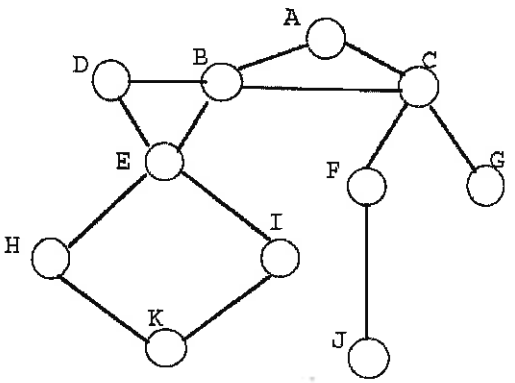
(a) An array contains the elements shown below. What would be the value of the elements in the array after each pass until the array is sorted using **insertion sort algorithm**.

[3 marks]

29	10	14	37	13	5

(b) For the following graph, list the vertices in order of a **depth-first search** starting at **vertex *E***. Assume adjacent vertices are accessed in an increasing alphabetical order.

[2 marks]



(c) Given the hash function:

[2 marks]

$h(x) = x \bmod 5$

Assume that we perform insertions of search keys in the order 257, 122, 177, 84. Indicate the position of the data if we use **linear probing** to resolve collisions. Use the following hash table to illustrate your answer in the answer booklet.

hash table

[0]	[1]	[2]	[3]	[4]

Continued .....

(d) Below is the class declaration for a pointer implementation of Linked List. You have to implement a function `insert`, which will **insert data** at the specified node of the Linked List.

Complete the function `insert` by adding a code segment to insert new node at the beginning of the list, and another code segment to any other locations specified by the given index.

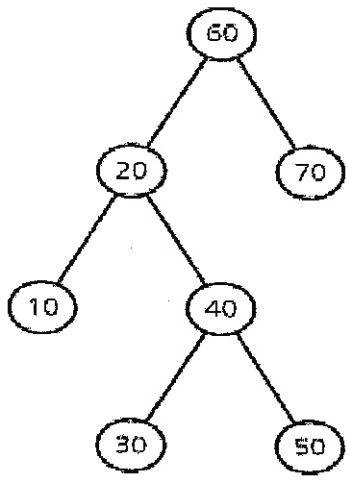
[3 marks]

```
class List
{
   public:
      ..........
      void insert(int index, const ListItemType& newItem);
      ..........
   private:
      struct ListNode
      {
        ListItemType item;
        ListNode *next;
      };
      int       size;
      ListNode *head;
      ListNode *find(int index) const;
};

void List::insert(int index, const ListItemType& newItem)
{
      int newLength = getLength() + 1;
      ..........
      ListNode *newPtr = new ListNode;
      newPtr->item = newItem;
      size = newLength;

      if (index == 1)
      {
            // insert new node at beginning of list
            // Write your codes in the Answer Booklet
      }
      else
      {     ListNode *prev = find(index-1);

            // insert new node after node to which prev
      points
            // Write your codes in the Answer Booklet
      }
}
```

Continued .....

Question 3

(a) Given the following binary search tree.



(i) What are the preorder, inorder, and postorder traversals of the binary search tree?

[2 marks]

(ii) Draw the final tree diagram after you insert the nodes 80, 65, 75, 45, 35, and 25, in that order?

[2 marks]

(b) What is the output of the following program?

[3 marks]

```
#include <iostream>
using namespace std;

void fun(int t)
{
 if(t > 0)
 {
   cout << "I love Data Structures and Algorithms.\n";
   fun(t-1);
   cout << "See you again\n";
 }
 else
  cout << "Good Bye !!!\n";
}
int main() {
    fun(3);
    system("PAUSE");
    return 0;
}
```

Continued .....

(c) Below is the class declaration for a pointer implementation of Linked List. You have to implement a function `remove`, which will **remove a specified node** of the Linked List.

Complete the function `remove` by adding a code segment to delete the first node from the list, and another code segment to delete a node from any other locations specified by the given index.

[3 marks]

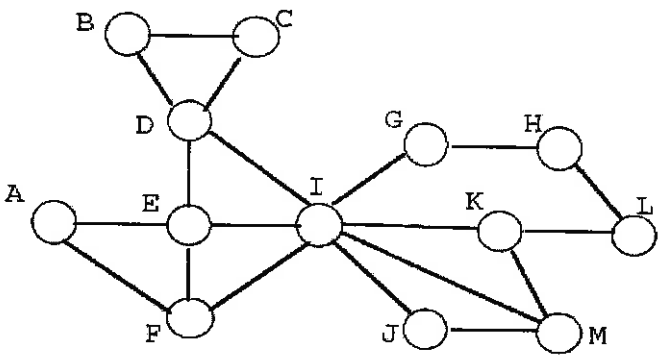
```
class List
{
   public:
      ..........
      void remove(int index);
      ..........
   private:
      struct ListNode
      {
         ListItemType item;
         ListNode    *next;
      };
      int       size;
      ListNode *head;
      ListNode *find(int index) const;
};

void List::remove(int index)
{
      ListNode *cur;
      --size;
      if (index == 1)
      {
            // delete the first node from the list
            // Write your codes in the Answer Booklet
      }
      else
      {     ListNode *prev = find(index - 1);

            // delete the node after the node to which prev
               points
            // Write your codes in the Answer Booklet
      }
      cur->next = NULL;
      delete cur;
      cur = NULL;
}
```

Continued .....

Question 4

(a) When do you need to apply Backtracking? Explain. [2 marks]

(b) For the following graph, list the vertices in order of a **breadth-first search** starting at **vertex *E***. Assume adjacent vertices are accessed in an increasing alphabetical order. [2 marks]



(c) Below is the class declaration for a pointer implementation of Stack. You have to implement a function `push`, which will **insert a specified node** to the Stack. Complete the function `push` by adding a code segment to set the data portion of the new node, and another code segment to insert the new node to the stack. [3 marks]

```
class Stack
{     public:
      ..........
       void push(const StackItemType& newItem);
      ..........
      private:
      struct StackNode
      {
            StackItemType item;
            StackNode    *next;
      };
      StackNode *topPtr;
};

void Stack::push(const StackItemType& newItem)
{
       StackNode *newPtr = new StackNode;

       // set data portion of new node
       // Write your codes in the Answer Booklet

       // insert the new node
       // Write your codes in the Answer Booklet
}
```

Continued .....

(d) The following program sorts integer numbers stored in an array "A" in ascending order and display them. This program uses "**Selection Sort**" algorithm to sort integer numbers. You have to complete the program to produce the desired result.

[3 marks]

```
#define MAX 5

int Min(int a[], int pos)
{
    int i,m,index=pos;
    m=a[pos]; //Get the Current Value
    for (i=pos+1;i<MAX;i++)//Search from next
        if (a[i]<m)
        {
          // Write your codes in the Answer Booklet
        }
    return index;
}

int main()
{
    int A[MAX],i,temp,k;
    cout<<"Enter"<<MAX<<" elements to be sorted:"<<endl;
    for (i=0;i<MAX;i++)
        cin>>A[i];
    for (i=0;i<MAX;i++)
    {
        k=Min(A,i);     //Find
        if (A[k]<A[i]) //Swap
        {
          // Write your codes in the Answer Booklet

        }
    }
    cout<<"The elements after Selection Sort"<<endl;
    for (i=0;i<MAX;i++)
        cout<<A[i]<<endl;
}
```

End of Page